National University

of Computer and Emerging Sciences, CFD Campus

AI 2002 - Artificial Intelligence (Spring 2023)
Assignment # 3

Topics Covered: Local search, adversarial search, constraint satisfaction problem	Submission Deadline: **April 03, 2023, by 16.00 sharp** Submit your source files in a zipped folder on the Google Classroom (of your respective section)

Problem # 1: [Local Search Algorithms]

Solve the N-queens problem using the following two local search algorithms: (a) Hill Climbing, and, (b) Simulated Annealing. Use the following to implement your algorithms.

- **Program input:** take no. of queens (N) as input from the user (any value between 5 to 10)
- **Board configuration:** on an NxN board, queens will be represented by the letter ‘Q’, and empty cells will be represented as a ‘dash’ (-) character
- **Start state:** random placement of N queens on an NxN board (each in one column)
- **Successor function:** a successor or neighbor of a state can be generated by randomly moving any one of the Queen inside that column
- **Objective function to minimize:** no. of queens attacking each other (commutative: Q1 attacking Q4 and vice verse is just one attack)
- **Temperature function for Simulated Annealing:** use the following link as a reference to find different temperature functions (https://mlrose.readthedocs.io/en/stable/source/decay.html), use *ExpDecay* in your algorithm
- **Program output:** to compare the performance of the two algorithms, make sure that both algorithms begin with the same start state. As both algorithms converge, print the following as output.
  - Success or Failure (i.e., goal state reached or not)?
  - No. of moves made
  - Which algorithm achieves the best minima?

Problem # 2: [Adversarial Search]

Implement the Tic-Tac-Toe game for two rational agents (Max and Min) who will play adversaries to each other using the Minimax algorithm. If the two rational agents play against each other in a zero-sum game, who will win actually?

Given the following problem description, implement the following algorithms.

(a) Minimax algorithm (without pruning)

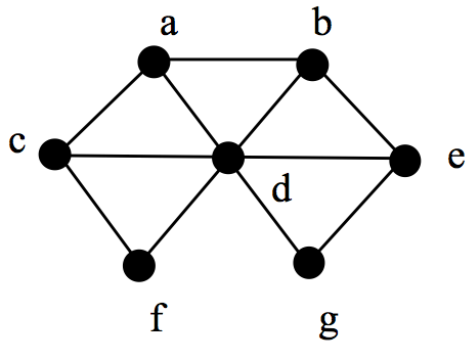
(b) Minimax with the Alpha-Beta pruning algorithm and show the level of the node which was pruned (assume that the start state is at level 0)

- **Random toss:** declare who won the toss, Max or Min. The player who won the toss will take the first move
- **Players moves:** represent Max move as an 'X' and Min move as an 'O'
- **Start state:** the 3x3 board is empty initially
- **Successor function:** no. of empty cells that can be filled by a player
- **Terminal states with utility values:** Max wins (+1 utility), Min wins (-1 utility), Draw (0 utility)
- **Program output:**
    - Board configuration after each player's move
    - Announce the winning player
    - No. of moves taken by each player as the game ends
    - Total no. of successor nodes generated by each player during search

Problem # 3: [Constraint Satisfaction Problem]

Given an undirected graph, we want to solve the graph coloring problem as described below.

- **Variables:** a, b, c, d, e, f, g
- **Domain:** R, G, B
- **Constraint:** any two neighbors cannot have the same color



Implement a solution to this problem using the following and use chronological order to select the next variable to break a tie or if no heuristic is used.

(a) Solve the problem using AC-3.

(b) Solve the problem using DFS with backtracking and heuristics including MRV, DH, and, LCV.

Program output:

- Show the values of all variables after each assignment
- Total number of assignments made to solve the problem
- For (b), count the number of backtracking performed and the no. of calls made to each heuristic