Oscar Galindo and Daniel Almeraz

HW 1

1. Give a complete problem formulation for each of the following. Choose a formulation that is precise enough to be implemented.
   1. Using only four colors, you have to color a planar map in such a way that no two adjacent regions have the same color.
      1. Initial State: All regions in the map are uncolored
      2. States: List of possible colored-regions configurations.
      3. Actions: Color region, change color, change color of region, move to adjacent region, check adjacent.
      4. Transition Model: Move to an uncolored region and choose a non-adjacent color.
      5. Goal Test: The map fully colored and all adjacent regions do not share the same color.
      6. Path Cost: Number of Actions (i.e. change color,
   2. A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. He would like to get the bananas. The room contains two stackable, moveable, climbable 3-foot-high crates.
      1. Initial State: The monkey is standing on the floor, the crates are next to each other, unstacked.
      2. States: The list of possible objects positions-including the monkey.
      3. Actions: Stack or Unstack, move crate, climb crate(s), move monkey, grab banana.
      4. Transition Model: Change the configuration of objects in the room-including monkey.
      5. Goal Test: The number bananas left is equal to zero.
      6. Path Cost: Number of Actions.
   3. You have a program that outputs the message “illegal program record” when fed a certain file of input records. You know that the processing of each record is independent of the other records. You want to discover what record is illegal.
      1. Initial State: Reading Input records.
      2. States: Reading, validating.
      3. Actions: Read, reject, accept, print error.
      4. Transition Model: A file is read and accepted, or a file is read and rejected.
      5. Goal Test: Illegal record was found
      6. Path Cost: Number of records processed before error “illegal program record is found”.
   4. You have three jugs, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure out exactly one gallon.
      1. Initial State: All of the jugs are empty
      2. States: All the possible combination of filled and empty jugs there are.
      3. Actions: Fill a jug, empty a jug, transfer, measure.
      4. Transition Model: The amount of water changes or its location changes.
      5. Goal Test: Measure up to one gallon.
      6. Path Cost: Number of actions taken to measure one gallon.

2) (Text 3.14) Which of the following are true and which are false? Explain your answers.

1. Depth-first search always expands at least as many nodes as A\* search with an admissible heuristic.
   * False, Depth-search could find a solution because of the probabilistic opportunity there is of finding optimal solutions easily and A\* search will have to exhaust all non-optimal paths compare to the optimal to prove and return a goal path.
2. h(n) = 0 is an admissible heuristic for the 8-puzzle.
   * True, admissibility is determined by the heuristic being not an overestimate of the cost and non-negative.
3. A\* is of no use in robotics because percepts, states, and actions are continuous.
   * It is of use if the continuous readings are separated as discrete cases, nevertheless in a pure continuous instance it would be inefficient and probably unfeasible.
4. Breadth-first search is complete even if zero step costs are allowed.
   * Assuming unlimited yes, it complete because the cost is not preventing the algorithm of exploring alternatives.
5. Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.
   * Manhattan distance applies to only to one-step movements whereas rooks can move from one extreme to the other in a single step.

3) Consider the search space below, where S is the start node and G1 and G2 satisfy the goal test. Arcs are labeled with the cost of traversing them and the estimated cost to a goal is reported inside nodes (so lower scores are better).

1. Best-first search
   1. S evaluated
   2. Assess if at goal? No.
   3. Expand and sort from most desirable to least desirable: B, D, A
   4. Visit most desirable: B
   5. Assess if at goal? No.
   6. Expand Most desirable and sort by desirability: G1, C
   7. Visit most desirable: G1
   8. Assess if at goal? Yes.
   9. Return Goal Path
2. Iterative Deepening
   1. Expand Level 1: S
   2. Is any path ending at goal? No.
   3. Increase depth
   4. Expand up to level 2: A|B|D
   5. Is any path ending at goal? No.
   6. Increase depth
   7. Expand up to level 3: C|G1|C|E|B|S
   8. Is any path ending at goal? Yes – G1.
   9. Return Path: S|B|G1
3. A\* Search
   1. Visit S and calculate heuristic + cost
   2. Evaluate heuristic + path cost for all possible alternatives: SA (cost 11), SB (cost 10), and SD (cost 9).
   3. Expand to smallest: D
   4. Evaluate heuristic + path cost for all possible alternatives: SA (cost 11), SB (cost 10), SDE (cost 13), SDS (20), SDC (10)
   5. Expand the most promising path: SB (10)
   6. Evaluate heuristic + path cost for all possible alternatives: SA (cost 11), SDE (cost 13), SDS (20), SDC (10), SBC (12), SBG1(16)
   7. Expand the most promising path: SDC (10)
   8. Evaluate heuristic + path cost for all possible alternatives: SA (11), SDE ( 13), SDS (20), SBC (12), SBG1(16), SBCJ (16), SBCS (18), SBCF (15).
   9. Expand the most promising path: SA (11)
   10. Evaluate heuristic + path cost for all possible alternatives: SDE (13), SDS (20), SBC (12), SBG1(16), SBCJ (16), SBCS(18), SBCF(15), SAB(9).
   11. Expand the most promising path: SAB (9)
   12. Evaluate heuristic + path cost for all possible alternatives: SDE (13), SDS (20), SBC (12), SBG1(16), SBCJ (16), SBCS (18), SBCF (15), SABC (11), SABG1(15).
   13. Expand the most promising path: SABC (11)
   14. Evaluate heuristic + path cost for all possible alternatives: SDE (13), SDS (20), SBC (12), SBG1(16), SBCJ (16), SBCS (18), SBCF (15), SABG1(15), SABCF (14), SABCS (17), SABCJ (15).
   15. Expand the most promising path: SBC (12)
   16. Evaluate heuristic + path cost for all possible alternatives: SDE (13), SDS (20), SBG1(16), SBCJ (16), SBCS (18), SBCF (15), SABG1(15), SABCF (14), SABCS (17), SABCJ (15) SBCJ(16) SBCS SBCF.
   17. …. (All paths less than SDCFG2(14) are not extended.
   18. Final Path at SDCFG2(14)
   19. Return Final Path.